IFW
AF
2126

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Bahrs et al.**

Serial No.: **09/430,814**

Filed: **October 29, 1999**

For: **Method and Apparatus in a Data Processing System for the Issuance and Delivery of Lightweight Requests to Concurrent and Multiple Service Providers**

35525

PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

§ Group Art Unit: **2126**
§
§ Examiner: **Nguyen, Van H.**
§
§ Attorney Docket No.: **AUS990339US5**
§

Certificate of Mailing Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being deposited with the United States Postal Service as First Class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on September 8, 2004.

By: Amelia C. Turner
Amelia C. Turner

TRANSMITTAL DOCUMENT

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:
ENCLOSED HEREWITH:

- Appellant's Brief (in triplicate) (37 C.F.R. 1.192); and
- Our return postcard.

A fee of $330.00 is required for filing an Appellant's Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

Respectfully submitted,

Duke W. Yee
Registration No. 34,285
YEE & ASSOCIATES, P.C.
P.O. Box 802333
Dallas, Texas 75380
(972) 367-2001
ATTORNEY FOR APPLICANTS

Docket No. AUS990339US5 *PATENT*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Bahrs et al.**

Serial No.: **09/430,814**

Filed: **October 29, 1999**

For: **Method and Apparatus in a Data Processing System for the Issuance and Delivery of Lightweight Requests to Concurrent and Multiple Service Providers**

Group Art Unit: **2126**

Examiner: **Nguyen, Van H.**

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

ATTENTION: Board of Patent Appeals and Interferences

Certificate of Mailing Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being deposited with the United States Postal Service as First Class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on 09.08.04.

By: Amelia C. Turner

09/15/2004 HALI11 00000064 090447 09430814
01 FC:1402 330.00 DA

APPELLANT'S BRIEF (37 C.F.R. 1.192)

This brief is in furtherance of the Notice of Appeal, filed in this case on July 8, 2004.

The fees required under § 1.17(c), and any required petition for extension of time for filing this brief and fees therefore, are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

This brief is transmitted in triplicate. (37 C.F.R. 1.192(a))

REAL PARTIES IN INTEREST

As reflected in the Assignment recorded on October 29, 1999, at Reel 010358, Frame 0125, the present application is assigned to International Business Machines Corporation, the real party in interest.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

Claims 124-144 stand finally rejected as noted in the Final Office Action mailed April 8, 2004.

STATUS OF AMENDMENTS

Applicants' Response to Office Action, transmitted on January 14, 2004, has been entered. Applicants have not filed a response to the Final Office Action.

SUMMARY OF INVENTION

Claim 124 describes a method for managing requests. A view controller object handles a container. The view controller object sends a view event to an application mediator object where the view event describes an action on the container. The application mediator object created this view controller object.

The application mediator sends a request event to a transporter object in response to receiving the view event from the view controller object that the application mediator created.

The transporter object then receives this request event from the application mediator. The request event received from the application mediator is self-identifying. The request event includes a type, a major code that identifies a class name of a destination object, a minor code that identifies a method name to be invoked, and includes object data.

The destination object is not included in the transporter object.

The transporter object then identifies the destination object identified in the request event and sends the request event to the identified destination object.

Claim 128 depends from claim 124 and further describes the request event including an indication to access a service at a remote location. The destination object accesses the service at the remote location in response to receiving the request event. The destination object also formats the request event into a form that is recognizable by the remote location.

Claim 129 depends from claim 128 and further describes processing the request event by the service, formatting a response into a new request event, and returning the new request event back to the transporter object.

The remaining claims are similar in scope to those discussed above.

ISSUES

Is the Examiner's rejection of claims 124-144 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent 6,275,232 issued to *Cataudella* well founded?

GROUPING OF CLAIMS

For the purposes of this appeal, claims 124-144 stand or fall together as one group.

ARGUMENT

The Examiner rejected claims 124-144 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent 6,275,232 issued to *Cataudella*. This position is not well founded.

Cataudella describes a method and system for zooming in a graphical user interface. *Cataudella* provides an event manager, a view manager, zooming objects, and a pick handler. Although *Cataudella* does use some of the same terms as used by Applicants in Applicants' claims, the elements of *Cataudella* are not used in the manner claimed by Applicants. Therefore, it is very important to carefully consider how the various elements of *Cataudella* interact with each other.

Cataudella teaches a zooming engine that includes a universe module 408. Module 408 includes a view manager module 410, an object manger module 412, and an event manger module 414. These modules may also be referred to as objects or managers.

Events 430 are received by the event manager 414. Examples of such events 430 are mouse and keyboard events initiated by a user. The event manager 414 then produces event objects 426 responsive to these received events 430. These event objects 426 are then sent to either the view manager 410 or a zooming object module 416. Event handling information is sent along with these event objects. The view manager 410 and zooming object 416 implement one or more event handlers for handling the events.

Thus, event manager 414 receives an event 430 and responds to that event by creating an event object 426. The event object is assigned to a target object which is either the view manager 410 or a zooming object. The event object includes a field which identifies its target.

The target object, i.e. either the view manager or a zooming object, then processes the event object by accessing event handlers associated with the target. When the target object is the view manager, the view manager may process the event object by passing the event object to a zooming object. The view manager may also process the event object by generating a new event object that is sent back to the event manager 414.

Therefore, to summarize, events are received by the event manager which in response generates event objects. These event objects are then sent to either the view manager or directly to a zooming object. The view manager or zooming object includes event handlers that process

the event objects that are received. If the event object is sent to the view manager, the view manager can respond by sending the event object to a zooming object or by generating a new event object that is sent back to the event manager.

Cataudella does not teach a view controller as claimed by Applicants. Applicants' claims describe a view controller that handles a container. The view controller sends a view event that describes an action on the container to an application mediator. This application mediator created the view controller. Thus, the view controller sends a view event to an application mediator that created the view controller.

The application mediator then sends a request event, which is not the same thing as the view event, to a transporter object. The transporter object then receives this request event and uses the request event to identify a destination object. The request event is then forwarded to the destination object.

The view controller of Applicants' claims is not similar to the view manager of *Cataudella*. Applicants claim the view controller sending a view event to an application mediator that created the view controller. The view manager of *Cataudella* sends event objects to either a zooming object or generates a new event object that the view manager sends back to the event manager. The view manager of *Cataudella* does not send a view event to an application mediator that created the view manager. Neither the event manager nor a zooming object created the view manager. In fact, nothing in *Cataudella* describes another object creating the view manager.

The view controller of Applicants' claims is not similar to the event manager of *Cataudella*. Again, Applicants claim that the view controller is created by the application mediator. The event manager of *Cataudella* sends event objects to either the view manager or a zooming object. The event manager of *Cataudella* is not created by either the view manager or a zooming object.

Therefore, nothing in *Cataudella* teaches a view controller that is created by an application mediator. Neither the event manager nor the view manager of *Cataudella* is created by another object.

Further, Applicants do not merely claim that the view controller is created by the application mediator. Applicants claim that the view controller sends a view event to the

application mediator that created the view controller. Nothing in *Cataudella* teaches either the event manager or view manager sending an event object to a device that created either the event manager or the view manager.

Cataudella does not teach an application mediator as claimed by Applicants. As described above, nothing in *Cataudella* teaches an application mediator that creates a view controller. Further, Applicants also claim that the application mediator receives a view event and then responds by sending a request event, which is a different event, to a transporter object. Nothing in *Cataudella* teaches a device that receives a view event and then responds by sending a new type of event, i.e. a request event to a transporter object. The view manager of *Cataudella* receives an event object and then either forwards that same event object to a zooming object or responds by creating a new event object that it sends back to the event manager. The view manager of *Cataudella* does not receive a view event and then respond by sending a request event to a transporter object. The event manager of *Cataudella* is not a transporter object as claimed by Applicants.

Applicants claim the transporter object receiving the request event and then using it to identify a destination object. Nothing in *Cataudella* teaches the event manager receiving this new event object and then using the new event object to identify a destination object.

Applicants claim the request event being self-identifying by including a type, a major code that identifies a class name of a destination object, a minor code that identifies a method name to be invoked, and object data. The event objects sent by the event manager of *Cataudella* include a field which identifies a target. The target is either the view manager or a zooming object. Nothing in *Cataudella* teaches a request event that is sent from an application mediator that includes a type, a major code that identifies a class name of a destination object, a minor code that identifies a method name to be invoked, and object data.

Applicants claim several different objects. Applicants claim a container, a view controller, an application mediator, a transporter object, and a destination object. Applicants also describe in detail how these various objects interact. *Cataudella* does not teach all of these objects. Although *Cataudella* does describe a view manager and an event manager, these managers are not the same as any of the objects claimed by Applicants. *Cataudella* does not teach a view controller, an application mediator, a transporter object, or a destination object as

claimed by Applicants. Further, *Cataudella* does not teach objects interacting in the manner claimed by Applicants.

The Examiner appears to rely on the same view manager and event manager in *Cataudella* to teach all of these objects claimed by Applicants. The view manager and event manager, however, can not be said to teach all of these objects claimed by Applicants. Further, the view manager and event manager do not interact with one another as claimed by Applicants.

Applicants' claim 128 further describes the request event including an indication to access a service at a remote location. The destination object accesses the service at the remote location in response to receiving the request event. The destination object also formats the request event into a form that is recognizable by the remote location. *Cataudella* does not teach a service at a remote location. The processing in *Cataudella* occurs either at the view manager or zooming object. Neither the view manager nor the zooming object is a remote location such as claimed by Applicants.

Applicants' claim 129 depends from claim 128 and further describes processing the request event by the service, formatting a response into a new request event, and returning the new request event back to the transporter object. Nothing in *Cataudella* describes processing the request event that was sent to the transporter object and then forwarded by the transporter object to a destination. Nothing in *Cataudella* teaches returning a new request event back to the transporter object. The only part of *Cataudella* that teaches generating a new event object is the part that describes the view manager sending a new event object back to the event manager. As described above, the event manager is not a transporter object such as claimed by Applicants. Therefore, *Cataudella* does not teach the features of this claim.

Cataudella does not describe, teach, or suggest a view controller, an application mediator, a transporter object, or a destination object as claimed by Applicants. Further, *Cataudella* does not describe, teach, or suggest objects interacting in the manner claimed by Applicants. Therefore, Applicants' claims are patentable over the cited prior art.

Lisa L.B. Yociss
Reg. No. 36,975
Yee & Associates, P.C.
PO Box 802333
Dallas, TX 75380
(972) 367-2001

APPENDIX OF CLAIMS

The text of the claims involved in the appeal reads:

124. A method in an object oriented data processing system for managing requests, the method comprising the data processing system implemented steps of:

sending a view event from a view controller object to an application mediator object that created the view controller object, said view event describing an action on a container, the container is handled by the view controller object;

responsive to a receipt of the view event by the application mediator object, sending a request event from the application mediator object to a transporter object;

receiving the request event at said transporter object, the request event being self identifying by including within the request event a type, a major code that identifies a class name of one of a plurality of destination objects, a minor code that identifies a method name to be invoked, and object data;

said plurality of destination objects not being included in said transporter object;

identifying, by said transporter object, said one of said plurality of destination objects; and

sending the request event to the identified one of the plurality of destination objects.

125. The method of claim 124, wherein the one of the plurality of destination objects accesses a service.

126. The method of claim 125, wherein the service is located on a remote data processing system.

127. The method of claim 124, further comprising:

formatting, by the one of the plurality of destination objects, the request event into a form that is recognizable by a destination.

128. The method of claim 124, further comprising:

said request event including an indication to access a service at a remote location;

responsive to receiving the request event at the one of the plurality of destination objects, accessing the service at the remote location using the one of the plurality of destination objects; and

formatting, by the one of the plurality of destination objects, the request event into a form that is recognizable by the remote location in order to access the service.

129. The method of claim 128 further comprising:

processing the request event by the service;

receiving a response to said request event from the service after said service has processed said request event;

formatting the response into a new request event; and

returning the new request event to the transporter object.

130. The method of claim 129, further comprising:

said request event requesting data; and

the new request event including the requested data.

131. The method of claim 129, wherein the remote service is a database.

132. An object oriented data processing system comprising:

a transporter object, responsive to receiving a request event from an application mediator object, the transporter object identifying one of a plurality of destination objects using a first indication that is included within said request event, said plurality of destination objects not being included in said transporter object;

the application mediator object creating a view controller object that sends view events from the view controller object to the application mediator object, the view events describing actions on a container, the container is handled by the view controller object;

the transporter object routing the request event to the one of the plurality of destination objects; and

the one of the plurality of destination objects performing a function on the request event, the function being identified from a second indication that is included within the request event.

133. The data processing system of claim 132, wherein the first indication is a major code and the second indication is a minor code.

134. The data processing system of claim 133, wherein the major code is a class name of the one of the plurality of destination objects and the minor code is a method name that is to be invoked.

135. The data processing system of claim 132, wherein the request event includes data.

136. An object oriented data processing system for managing requests, the data processing system comprising:

a view controller object for sending a view event from the view controller object to an application mediator object that created the view controller object, the view event describing an action on a container, the container is handled by the view controller object;

the application mediator sending a request event to a transporter object responsive to a receipt of the view event by the application mediator object;

said transporter object for receiving the request event, the request event being self identifying by including within the request event a type, a major code that identifies a class name of one of a plurality of destination objects, a minor code that identifies a method name to be invoked, and object data;

said plurality of destination objects not being included in said transporter object;

the transporter object identifying said one of said plurality of destination objects using the request event; and

sending means for sending the request event to the identified one of the plurality of destination objects.

137. The data processing system of claim 136, wherein the one of the plurality of destination objects accesses a service.

138. The data processing system of claim 137, wherein the service is located on a remote data processing system.

139. The data processing system of claim 136, further comprising:

the one of the plurality of destination objects formatting the request event into a form that is recognizable by a destination.

140. The data processing system of claim 136, further comprising:

said request event including an indication to access a service at a remote location;

accessing means, responsive to receiving the request event at the one of the plurality of destination objects, for accessing the service at the remote location using the one of the plurality of destination objects; and

the one of the plurality of destination objects formatting the request event into a form that is recognizable by the remote location in order to access the service.

141. The data processing system of claim 140 further comprising:

the service for processing the request event;

second receiving means for receiving a response to the request event from the service after the service has processed the request event;

formatting means for formatting the response into a new request event; and

returning means for returning the new request event to the transporter object.

142. The data processing system of claim 141, further comprising:

said request event requesting data; and

the new request event including the data.

143. The data processing system of claim 141, wherein the remote service is a database.

144. A computer program product in a computer readable medium for use in an object oriented data processing system for managing requests, the computer program product comprising:

instructions for sending a view event from a view controller object to an application mediator object that created the view controller object, said view event describing an action on a container, the container is handled by the view controller object;

instructions responsive to a receipt of the view event by the application mediator object, for sending a request event from the application mediator object to a transporter object;

instructions for receiving the request event at the transporter object, the request event being self identifying by including within the request event a type, a major code that identifies a class name of one of a plurality of destination objects, a minor code that identifies a method name to be invoked, and object data;

said plurality of destination objects not being included in said transporter object;

instructions for identifying, by said transporter object, one of said plurality of destination objects; and

instructions for sending the request event to the identified one of the plurality of destination objects.